Evolutionary Computation of Forests with Degree- and Role-constrained Minimum Spanning Trees

Laura Anton-Sanchez L.ANTON-SANCHEZ@UPM.ES
Concha Bielza MCBIELZA@FI.UPM.ES
Pedro Larrañaga PEDRO.LARRANAGA@FI.UPM.ES
Departamento de Inteligencia Artificial
Universidad Politecnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Madrid, Spain

Abstract

Finding the degree-constrained minimum spanning tree (DCMST) of a graph is a widely studied NP-hard problem. One of its most important applications is network design. Here we deal with a new variant of the DCMST problem, which consists of finding not only the degree- but also the role-constrained minimum spanning tree (DRCMST), i.e., we add constraints to restrict the role of the nodes in the tree to root, intermediate or leaf node. Furthermore, we do not limit the number of root nodes to one, thereby, generally, building a forest of DRCMSTs. The modeling of network design problems can benefit from the possibility of generating more than one tree and determining the role of the nodes in the network. We propose a novel permutation-based representation to encode these forests. In this new representation, one permutation simultaneously encodes all the trees to be built. We simulate a wide variety of DRCMST problems which we optimize using eight different evolutionary computation algorithms encoding individuals of the population using the proposed representation. The algorithms we use are: estimation of distribution algorithm, generational genetic algorithm, steady-state genetic algorithm, covariance matrix adaptation evolution strategy, differential evolution, elitist evolution strategy, non-elitist evolution strategy and particle swarm optimization. The best results are for the estimation of distribution algorithms and both types of genetic algorithms, although the genetic algorithms are significantly faster.

Keywords: Degree- and role-constrained minimum spanning tree, evolutionary computation, forest, network design, permutation problems.

1. Introduction

A spanning tree is a basic topological structure in network design problems such as transportation, telecommunications and distribution systems. Well-known-classical algorithms exist for building a minimum spanning tree (MST) (Kruskal, 1956; Prim, 1957). In practice a more realistic representation for network design is a degree-constrained minimum spanning tree (DCMST), i.e., an MST with constraints on the number of edges incident to each node. The DCMST problem can be applied in a transportation system, such as wires, pipes or canals, where the length of the connections of m nodes should be minimum. The handling capacity of each node imposes a constraint on the number of edges that can be

connected to that node. In communication networks, the degree constraint limits network vulnerability if a node fails. The DCMST problem could also be applied to the design of a computer network or a road network with a maximum number of roads at a crossing (Krishnamoorthy et al., 2001).

The DCMST problem is NP-hard (this can be shown by reduction to the Hamiltonian path problem, Garey and Johnson (1979)). The problem of finding the DCMST of a graph, and particularly finding a good representation of the tree, has been widely studied in the literature. For example, Knowles et al. (2000) introduce the randomized primal method, a novel tree construction algorithm for stochastic iterative search techniques. This method builds low-cost degree-constrained trees. Krishnamoorthy et al. (2001) compare three heuristics (simulated annealing, a genetic algorithm and a method based on problem space search) and two exact algorithms (Lagrangian relaxation and branch and bound) for the DCMST problem. Further, they propose alternative tree representations to facilitate the genetic algorithm neighborhood searches. Raidl and Julstrom (2003) propose representing spanning trees for network design problems directly as sets of their edges. They demonstrate the usefulness of their encoding for the DCMST problem. Soak et al. (2004) develop another effective encoding method of a tree for use by black-box optimization methods.

The above representations are based on the construction of a single tree. Some more recent studies consider building a forest. This extension is not straightforward. In Delbem et al. (2004), the proposed forest representation, named node-depth encoding, is composed of the union of the encoding of all trees of the forest. The union is implemented using an array of pointers, where each pointer indicates a tree consisting of linear lists containing the tree nodes and their depths. The proposed approach is evaluated for the DCMST problem. Some years later, Delbem et al. (2012) propose a new data structure to generate and manipulate a set of spanning forests, called node-depth-degree representation. This structure improves the average running time of their previous node-depth encoding (the forest is again composed of the union of the trees). Also working with a group of trees, Czajko and Wojciechowski (2009) take a different approach. They formulate the hop- and degree-constrained minimum spanning forest problem with minimization of the number of trees (this problem is defined as part of an access network topology design).

Here we define another variation on the DCMST problem, which we call the degree- and role-constrained minimum spanning tree (DRCMST) problem. A DRCMST is a DCMST where we determine a priori the role of the nodes in the tree according to three alternatives: root node, intermediate node and leaf node. It may be useful to constraint the role of the nodes in network design. In computer networking, for example, the service has to reach the leaf nodes, and these nodes are clearly different from the central processor (which has a fixed number of ports). In such a network the cost could be associated with distances between nodes or with the material costs needed to connect nodes. A DRCMST could also be useful in a business network, for example, for the design of the project staff structure, where we would differentiate between the project manager (root), middle managers (intermediate nodes) and the rest of the team who are not in charge of any other staff (leaf nodes). The cost of this problem might be associated with team member preferences for project managers.

The DRCMST problem is also NP-hard, because it contains the particular case where we determine one root node and one leaf node with the constraint that the degree of each

node has to be less than or equal to two. This is equivalent to the shortest Hamiltonian path problem between two nodes. In addition, a forest rather than a single tree can be built, i.e., we do not limit the number of root nodes to one so we can solve more complex problems (e.g., we might design several computer networks and several business networks by simultaneously considering several central processors and several project managers in the above examples, respectively). In this paper, we introduce a new permutation-based representation for building forests of DRCMSTs. One permutation simultaneously encodes all the DRCMSTs in the forest. Due to problem complexity, we address a wide variety of DRCMST problems using evolutionary computation techniques. Individuals of the populations are encoded with the proposed representation.

The organization of this paper is as follows. Section 2 formally describes the DCMST and DRCMST problems. Section 3 introduces the proposed permutation-based representation to encode forests of DRCMSTs. This representation is used to approximate a wide variety of DRCMST problems using the eight evolutionary computation algorithms described in Section 4. Section 5 details the characteristics of the 60 simulated test problems used to compare the performance of the different evolutionary computation techniques on the problem of finding forests of DRCMSTs. Section 6 analyzes the results and compares the algorithms. Finally, some discussion and conclusions are provided in Section 7.

2. Problem definition

A DCMST is a spanning tree where we assume that there is a degree constraint on each node such that, at node v, its degree value $\deg(v)$ (i.e., its number of incident edges) is at most a given value $d_v \in \mathbb{N}$ and the total cost of the associated edges is minimum. Formally, let $G = (V, E)$ be an undirected complete graph with a set of vertices (nodes) V and a set of edges E. A spanning tree of G is a subgraph $T = (V, E_T)$, $E_T \subset E$ that contains all vertices in V which it connects with exactly $|V| - 1$ edges. Let $c_{uv} \geq 0$ be the cost of each edge $(u, v) \in E$, $u, v \in V$. The DCMST problem consists of finding a minimum spanning tree $T^* = (V, E_{T^*}), E_{T^*} \subset E$ such that

$$T^* = \underset{T}{\operatorname{argmin}} \sum_{(u,v) \in E_T} c_{uv},$$

subject to

$$\deg(v) \leq d_v \quad \text{for all } v \in V.$$

A DRCMST is a DCMST where the role of the nodes in the tree is determined a priori (by the user/expert). In a DRCMST problem, we define three subsets of nodes R, I and L for root nodes, intermediate nodes and leaf nodes, respectively, where each node $v \in V$ must belong to one and only one subset, $V = R \cup I \cup L$, $R \cap I = R \cap L = I \cap L = \varnothing$, $R \neq \varnothing$ and $L \neq \varnothing$. Note that the following conditions must be met: $d_v \geq 1 \ \forall v \in R$, $d_v \geq 2 \ \forall v \in I$ and $d_v = 1 \ \forall v \in L$. If we choose only one root node ($|R| = 1$), we construct a single tree. With a higher number of roots, we build a forest of DRCMSTs.

Given an undirected complete graph $G = (V, E)$ defined as above, a forest of G is a subgraph $F = (V, E_F)$, $E_F \subset E$ that contains all vertices in V and consists of a spanning

tree in each connected component of F. Given a definition of degree constraints and subsets R, I and L that satisfies the requirements specified in the previous paragraph, the DRCMST problem consists of finding a minimum forest $F^* = (V, E_{F^*}), E_{F^*} \subset E$ with $|R|$ connected components such that

$$F^* = \underset{F}{\operatorname{argmin}} \sum_{(u,v)\in E_F} c_{uv}, \tag{1}$$

subject to

$$\begin{gathered} \deg(v) \leq d_v \quad \text{for all } v \in V \\ \text{role}(v) = root \text{ for all } v \in R \\ \text{role}(v) = intermediate \text{ for all } v \in I \\ \text{role}(v) = leaf \text{ for all } v \in L, \end{gathered}$$

where $\text{role}(v) \in \{root, intermediate, leaf\}$ gives the role of node v in the forest.

Proposition 1. Given an undirected complete graph $G = (V, E)$, the subsets of nodes R, I and L such that $V = R \cup I \cup L$, $R \cap I = R \cap L = I \cap L = \varnothing$, $R \neq \varnothing$ and $L \neq \varnothing$, and a degree constraint for each $v \in V$ satisfying that $d_v \geq 1 \ \forall v \in R$, $d_v \geq 2 \ \forall v \in I$ and $d_v = 1 \ \forall v \in L$, the DRCMST problem is feasible if and only if

$$\sum_{v\in R} d_v + \sum_{v\in I} d_v - |I| \geq |I| + |L|. \tag{2}$$

Proof.

$\Rightarrow$

Suppose the problem is feasible, i.e., the forest of G consists of $|R|$ trees. Let us prove that (2) holds for $\deg(v)$, $v \in V$ and then it will also hold for $d_v, v \in V$, because $d_v \geq \deg(v)$, $\forall v \in V$. Then, we prove

$\sum_{v\in R} \deg(v) + \sum_{v\in I} \deg(v) - |I| \geq |I| + |L| \Longleftrightarrow \sum_{v\in R} \deg(v) + \sum_{v\in I} \deg(v) \geq 2|I| + |L|$

We add $\sum_{v\in L} \deg(v)$ to both sides of (2):

$\sum_{v\in R} \deg(v) + \sum_{v\in I} \deg(v) + \sum_{v\in L} \deg(v) \geq 2|I| + |L| + \sum_{v\in L} \deg(v)$

It is known that $\sum_{v\in V} \deg(v) = 2|E|$ holds for any graph $G = (V, E)$. Further, because $d_v = \deg(v) = 1 \ \forall v \in L$, $\sum_{v\in L} \deg(v) = |L|$ and we have

$\sum_{v\in V} \deg(v) \geq 2|I| + |L| + |L| \Longleftrightarrow 2|E| \geq 2|I| + 2|L| \Longleftrightarrow |E| \geq |I| + |L|$

Since $|E_T| = |V_T| - 1$ holds for every tree T and the forest has $|R|$ trees $\Rightarrow |E| = |V| - |R|$. Then

$|V| - |R| \geq |I| + |L|$ and this becomes an equality because $|V| = |R| + |I| + |L|$. Hence, because (2) is true for $\deg(v)$, $v \in V$, it is also true for $d_v, v \in V$.

$\Leftarrow$

Suppose that (2) holds. Let us prove that the problem is feasible, i.e., that we can build $|R|$ trees.

We add $\sum_{v\in L} d_v$ to both sides of (2) and arguing as in the previous case we have:

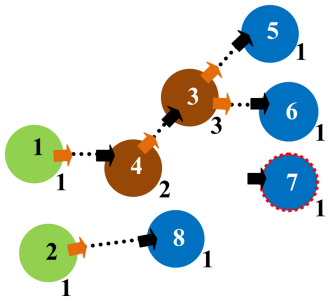


Figure 1: Example of an infeasible DRCMST problem. The maximum allowed degree d_v is shown on the right of each node. Root nodes are shown in green, intermediate nodes in brown and leaf nodes in blue. Since $|R| = 2$, $|I| = 2$ and $|L| = 4$, we need six "inputs" (black arrows): two for intermediate nodes and four for leaf nodes. However, we only have five possible "outputs" (orange arrows): two from the root nodes and three from intermediate nodes. In this example, node number 7 cannot be connected to either of the two trees in this forest. This example does not satisfy (2): $2 + 5 - 2 \not\geq 2 + 4$.

$\sum_{v\in R} d_v + \sum_{v\in I} d_v + \sum_{v\in L} d_v \geq 2|I| + |L| + \sum_{v\in L} d_v \iff \sum_{v\in V} d_v \geq 2|I| + |L| + |L| \iff 2|E| \geq 2|I| + 2|L| \iff |E| \geq |I| + |L|$

Because $|V| = |R| + |I| + |L| \Rightarrow |I| + |L| = |V| - |R|$, we have that $|E| \geq |V| - |R|$. To build $|R|$ trees it is necessary that $|E| = |V| - |R|$, hence the problem is feasible. ■

In other words, the DRCMST problem is feasible if and only if the maximum allowed number of "outputs" (left-hand side of (2)) is greater than or equal to the number of "inputs" (right-hand side of (2)). See Fig. 1 for a counterexample. Notice that we are working with undirected graphs so no input or output edges exist. By establishing root nodes, however, tree structure is implicitly directed since the roots (leaves) are considered to be the origin (end) of a tree.

3. Problem representation

We set out to encode the DRCMST problem using a permutation representation. A permutation is understood as a vector $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_n)$ of the indexes $1, ..., n$ such that $\sigma_k \neq \sigma_s$ for all $k \neq s$. We say that index s is in position k in $\boldsymbol{\sigma}$ when $\sigma_k = s$.

In a forest encoded by the proposed representation, all nodes have a degree $\deg(v)$ equal to their maximum allowed degree d_v. To verify this constraint, we add a new type of nodes called dummy nodes, see Fig. 2. We add as many dummy nodes as are necessary to make $\deg(v) = d_v$, $\forall v \in V$. Let D be the subset of dummy nodes. In a forest of DRCMSTs encoded by our representation, (2) becomes an equality:

$$\sum_{v\in R} d_v + \sum_{v\in I} d_v - |I| = |I| + |L| + |D|, \tag{3}$$

and hence the number of dummy nodes to be added is

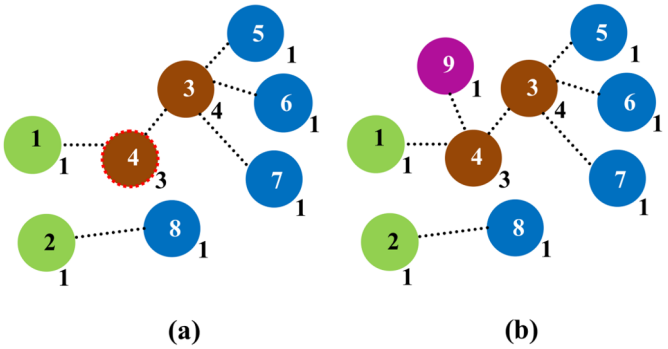


Figure 2: Example of a DRCMST forest with two trees. The maximum allowed degree d_v is shown on the right of each node. In (a) the degree $\deg(v)$ of all nodes is equal to their maximum allowed degree d_v, except node number 4 where $\deg(4)=2$ and $d_4=3$. To encode this forest with our permutation-based representation, we add one dummy node, node 9, connected to node 4. Forests (a) and (b) are equivalent because dummy nodes are added for representation purposes only and do not affect the calculation of tree costs.

$$|D| = \sum_{v \in R} d_v + \sum_{v \in I} d_v - 2|I| - |L|. \tag{4}$$

Dummy nodes are added for representation purposes only, and they are all leaf nodes so their degree is always equal to 1. The cost of every edge that reaches a dummy node is zero. Then, $m = |V| + |D| = |R| + |I| + |L| + |D|$ is the total number of nodes in the encoded forest.

In our representation, each index of the permutation denotes a connection between two nodes, i.e., each index represents an edge in the forest. Since $|E_T| = |V_T| - 1$ holds for every tree T and we encode $|R|$ trees in one permutation, the permutation length n (total number of edges in the encoded forest) can be calculated as $n = m - |R|$.

The length of the permutation can also be obtained using (3):

$$n = \sum_{v \in R} d_v + \sum_{v \in I} d_v - |I| = |I| + |L| + |D|.$$

To find out which nodes are connected by the edges represented in each position of the permutation, we need two auxiliary arrays, *parent* and *child*, both of length n. These arrays remain unchanged for all permutations of the same problem. The *parent* auxiliary array represents the "outputs" of the edges in the forest. Since each root node has d_v "outputs" and each intermediate node has $(d_v - 1)$ "outputs", each root node appears d_v times for all $v \in R$ and each intermediate node appears $(d_v - 1)$ times for all $v \in I$ in the *parent* array. The *child* auxiliary array represents the "inputs" of the edges. All intermediate nodes, leaf nodes and dummy nodes have one "input", therefore the *child* array includes all nodes $v \in I \cup L \cup D$ once. With these arrays, our permutation is such that $\sigma_k = s$ represents that node $parent_s$ in the forest (note that we use the subscript to indicate the element in position s of the auxiliary array) is the parent of node $child_k$, see Fig. 3. A simple version of this

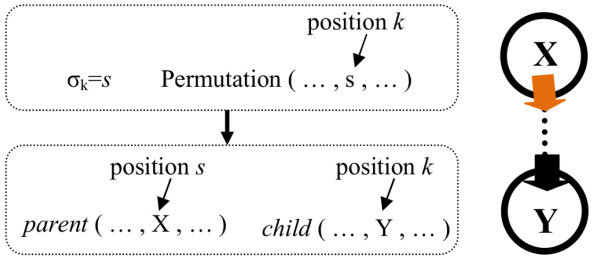


Figure 3: Decoding the proposed permutation-based representation. $\sigma_k = s$ represents that, in the forest, node $parent_s$ (node X) is the parent of node $child_k$ (node Y).

novel representation considering only binary trees was introduced in Anton-Sanchez et al. (2013).

To illustrate this representation, consider the example in Fig. 4. Fig. 4(a) shows the $|V| = 10$ nodes of an example graph $G = (V, E)$. The maximum allowed degree d_v is shown on the right-hand side of each node $v \in V$. Nodes selected as root nodes are shown in green, intermediate nodes in brown and leaf nodes are shown in blue, thus, $|R| = 2$, $|I| = 3$ and $|L| = 5$. This problem is feasible because it satisfies (2). We check if it is necessary to add any dummy nodes to solve the problem using (4):

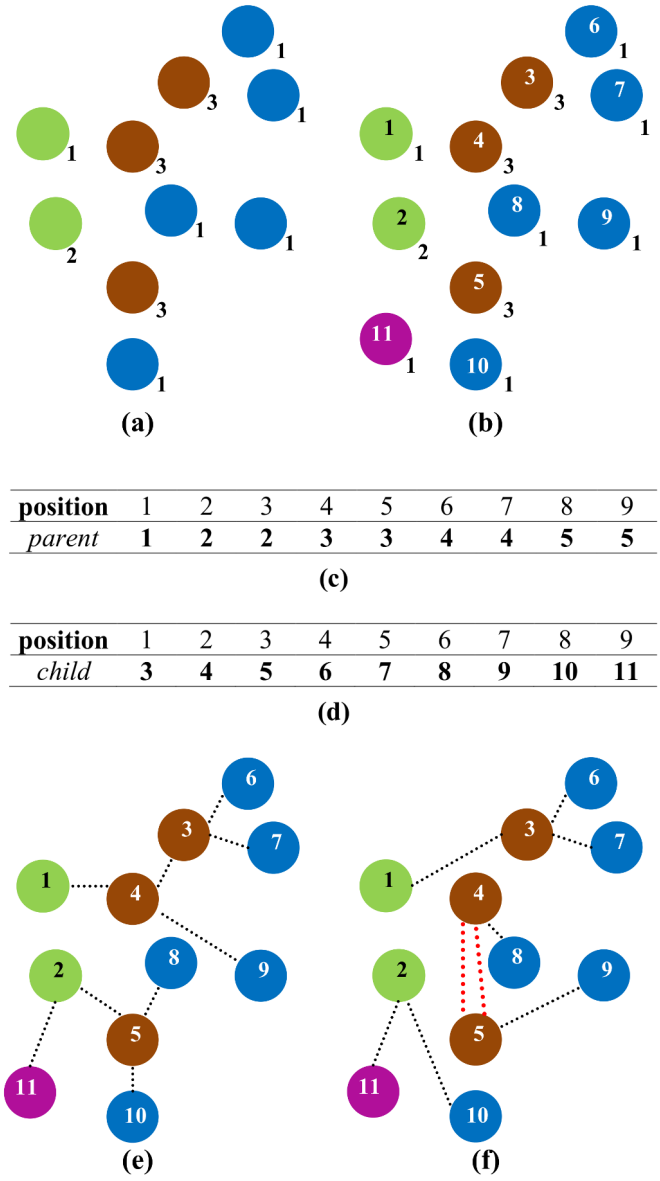
$$|D| = \sum_{v \in R} d_v + \sum_{v \in I} d_v - 2|I| - |L| = 3 + 9 - 2 \cdot 3 - 5 = 1,$$

i.e., we have to add one dummy node. Fig. 4(b) shows the numbered nodes and the added dummy node (node number 11 in pink). Then, a forest in the example will have two trees ($|R| = 2$) with $m = |V| + |D| = 10 + 1 = 11$ nodes and it will be represented by permutations of length $n = m - |R| = 11 - 2 = 9$.

We build the *parent* and *child* auxiliary arrays both needed to encode the permutations. As indicated, we add each root node d_v times ($v \in R$) and intermediate nodes $(d_v - 1)$ times each ($v \in I$) to the *parent* array. A possible *parent* auxiliary array is shown in Fig. 4(c). A possible *child* auxiliary array, including intermediate, leaf and dummy nodes once, is shown in Fig. 4(d). Note that *parent* and *child* auxiliary arrays must be established before starting to solve the problem because they determine which DRCMST problem solution each permutation represents. The order of the nodes in these arrays is in fact irrelevant.

Fig. 4(e) represents the permutation (6,1,2,4,5,8,7,9,3) which would be a correct individual (forest) using the defined *parent* and *child* auxiliary arrays, i.e., $parent_6$ (node 4) is the parent of $child_1$ (node 3), $parent_1$ (node 1) is the parent of $child_2$ (node 4) and so on until the last position of the permutation, which indicates that $parent_3$ (node 2) is the parent of $child_9$ (node 11).

Our permutation-based representation implicitly ensures that all constraints of problem (1) are satisfied in an encoded forest. However, permutations encoding any cycle represent invalid forests. For example, Fig. 4(f) represents an invalid individual (permutation (1,**8**,**6**,4,5,7,9,2,3)) because it contains a cycle (in red) between nodes 4 and 5. The second position of the permutation indicates that $parent_8$ (node 5) is the parent of $child_2$ (node 4) and the next position indicates that $parent_6$ (node 4) is the parent of $child_3$ (node 5). As



position	1	2	3	4	5	6	7	8	9
parent	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**

(c)

position	1	2	3	4	5	6	7	8	9
child	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

(d)

Figure 4: An example of permutation representation. (a) Nodes of an example graph. The maximum allowed degree d_v is specified to the right of each node. The role of each node is indicated in different colors: root nodes in green, intermediate nodes in brown and leaf nodes in blue. (b) Numbered nodes. According to (4) ($|D| = 3 + 9 - 2 \cdot 3 - 5 = 1$), a dummy node is needed to solve the problem. This is added as node 11 in pink. (c)-(d) *parent* and *child* auxiliary arrays required to determine which forest each permutation represents. (e) Example of correct individual, permutation (6,1,2,4,5,8,7,9,3). (f) Example of invalid individual because it contains a cycle, permutation (1,8,6,4,5,7,9,2,3).

we prove below, we can ensure that permutations whose representation contains no cycles are correct forests, i.e., they represent the required number of trees $|R| = m - n$.

Proposition 2. If a graph $G = (V, E)$ has m nodes, n edges and no cycles, then it is a forest composed of $(m - n)$ trees.

Proof. Suppose we have c connected components in G. Let n_k be the number of edges and m_k the number of nodes in component $k, k = 1, ..., c$. Since there are no cycles, these connected components are trees, and it holds for each one that:

$n_k = m_k - 1 \quad \forall k, k = 1, ..., c.$

Then,

$\sum_k n_k = \sum_k m_k - c \Longrightarrow n = m - c \Longrightarrow c = m - n,$

i.e., we have $(m - n)$ connected components, which are trees because they have no cycles, as we set out to prove. ■

Note that this representation yields several permutations encoding the same individual. For example, permutation (6,1,**3**,4,5,8,7,9,**2**) is the same individual as permutation (6,1,**2**,4,5,8,7,9,**3**) (Fig. 4(e)) because both $parent_3$ and $parent_2$ represent node number 2. We call these positions redundant positions, and we remove redundancy. To do this, we always choose the individual whose numbers of redundant positions are ordered from lowest to highest, i.e., (6,1,2,4,5,8,7,9,3) in the example.

Furthermore, note that cycles of length one, i.e., cycles that indicate that a node is its own parent, are very easy to detect with our representation. For example, as regards the problem illustrated in Fig. 4, we know that numbers 4 or 5 can not occupy the first position of the permutation because this would indicate that $parent_4$ or $parent_5$, i.e., node 3, is the parent of $child_1$, also node 3. For longer cycles, we must traverse the permutation and build the trees that it encodes to identify any cycles. To do this, we use the weighted quick-union algorithm (Sedgewick and Wayne, 2011). The worst-case order of growth of all operations of this algorithm is $\log n$, where n is the length of the permutation.

4. Problem-solving approach

We used eight different evolutionary algorithms to solve a variety of synthetic simulated problems and compare results. The algorithms analyzed were: estimation of distribution algorithm (EDA) (Larrañaga and Lozano, 2002), generational genetic algorithm (gGA) (Cobb and Grefenstette, 1993), steady-state genetic algorithm (ssGA) (Syswerda, 1991), covariance matrix adaptation evolution strategy (CMAES) (Hansen and Ostermeier, 1996), differential evolution (DE) (Storn and Price, 1997), elitist and non-elitist evolution strategy (ElitistES and NonElitistES) (Rechenberg, 1973) and particle swarm optimization (PSO) (Kennedy and Eberhart, 1995).

Genetic algorithms have been widely studied for solving permutation-based optimization problems (Larrañaga et al., 1999), and they are known to perform satisfactorily (Reeves, 1995; Ruiz and Maroto, 2005; Bielza et al., 2010). However, although several papers using probabilistic models on rankings with EDAs have recently been published (Ceberio et al., 2011; Aledo et al., 2013; Ceberio et al., 2014), EDAs have not been so extensively developed for permutation-based optimization problems (Ceberio et al., 2012). The probabilistic model learned in an EDA is expected to reflect the structure of the problem, and therefore this approach should provide a more effective exploitation of promising solutions than the crossover and mutation operators of genetic algorithms. For this reason, we were particularly interested in comparing EDA, gGA and ssGA.

The CMAES, DE, ElitistES, NonElisitisES and PSO were designed to solve real-value based problems but we also wanted to test their performance with the DRCMST problem. In order to apply these algorithms we used the random keys algorithm (Bean, 1994) by means of which we can encode a vector of real values in a permutation. A permutation can be obtained from a real vector $(x_1, x_2, ..., x_n)$ of length n by ranking the positions using the values x_i, $i = 1, ..., n$. For example, the real vector (5.12, 3.48, 10.21, 1.07, 0.75, 8.54) would represent the permutation (4, 3, 6, 2, 1, 5).

In order to compare the performance of these algorithms we used the jMetal framework (Durillo and Nebro, 2011). jMetal stands for Metaheuristic Algorithms in Java, and it is an object-oriented Java- based framework for single and multi-objective optimization with a variety of metaheuristics techniques. We found that jMetal contained all the algorithms in which we were interested, except the EDA. So, we added our own implementation of this technique to the framework. Given that the absolute position of each number in the permutation is closely related to performance in our problem (since each number in the permutation denotes a selected edge to build the forest), we implemented the node histogram sampling algorithm presented in Tsutsui (2006). This algorithm models number frequencies at each absolute position in individuals of a population.

For both the added EDA and the other single-objective algorithms already included in jMetal, we made improvements due to the specific characteristics of our representation. On the one hand, we ruled out the generation of individuals containing cycles of length one (which are easy to detect as described in Section 3) and, on the other hand, we removed the redundancy of our representation by selecting a representative individual from the redundant individuals as already explained.

We used the default parameters provided in jMetal for all the algorithms. For each problem, we established a population size equal to $10|V|$ for all algorithms. For each execution of each problem, the initial population was randomly generated including the improvements discussed above (length-one cycles and redundancy). We decided to stop any algorithm if there was no more than a 0.1% improvement of the best fitness over the last 500 generations.

5. Test problem generation

We simulated ten problems for each of the following sizes $|V|$= 20, 40, 60, 80, 100, 200 nodes. The number of roots and intermediate nodes were randomly generated, although some constraints were included. The number of problem root nodes was less than or equal to 20% of all problem nodes, i.e., $|R| \leq 0.2|V|$. The number of intermediate nodes was less than or equal to 75% of all problem nodes ($|I| \leq 0.75|V|$). The number of leaf nodes was derived as $|L| = |V| - |R| - |I|$. The maximum allowed degree of root and intermediate nodes (leaf nodes always have a degree equal to 1) was also simulated randomly for each node as follows. The maximum allowed degree d_v was between 1 and 4 for root nodes and between 2 and 5 for intermediate nodes.

We simulated two different types of problems: five problems for each size with Euclidean distances (Table 1) and five problems for each size with randomly simulated distances (Table 2). The set of problems with Euclidean distances was created by simulating coordinates (x, y, z) of each point between $x_{min} = y_{min} = z_{min} = 1$ and $x_{max} = y_{max} = z_{max} = 100$. Then, we computed the cost matrix with real Euclidean distances between pairs of points. For

Table 1: Description of the simulated problems with Euclidean distances. The table shows the number of root, intermediate, leaf and dummy nodes and the length of the permutations that encode the problem solutions.

Euclidean distances	$\|R\|$ **roots**	$\|I\|$ **intermediate**	$\|L\|$ **leaf**	$\|D\|$ **dummy**	n **permutation length**
Problem 1-20Nodes	3	9	8	10	27
Problem 2-20Nodes	3	6	11	0	17
Problem 3-20Nodes	1	9	10	3	22
Problem 4-20Nodes	3	11	6	10	27
Problem 5-20Nodes	2	12	6	11	29
Problem 1-40Nodes	5	18	17	15	50
Problem 2-40Nodes	7	12	21	6	39
Problem 3-40Nodes	4	22	14	16	52
Problem 4-40Nodes	4	18	18	8	44
Problem 5-40Nodes	7	16	17	13	46
Problem 1-60Nodes	5	24	31	1	56
Problem 2-60Nodes	1	36	23	13	72
Problem 3-60Nodes	11	17	32	8	57
Problem 4-60Nodes	9	26	25	18	69
Problem 5-60Nodes	11	23	26	22	71
Problem 1-80Nodes	13	42	25	50	117
Problem 2-80Nodes	11	25	44	1	70
Problem 3-80Nodes	13	36	31	36	103
Problem 4-80Nodes	12	33	35	5	73
Problem 5-80Nodes	15	29	36	21	86
Problem 1-100Nodes	15	45	40	29	114
Problem 2-100Nodes	7	56	37	39	132
Problem 3-100Nodes	6	52	42	14	108
Problem 4-100Nodes	18	42	40	39	121
Problem 5-100Nodes	15	56	29	49	134
Problem 1-200Nodes	23	74	103	24	201
Problem 2-200Nodes	30	100	70	91	261
Problem 3-200Nodes	11	102	87	43	232
Problem 4-200Nodes	10	103	87	16	206
Problem 5-200Nodes	20	99	81	59	239

all the problems with random distances, we computed the cost matrix by simulating costs between pairs of points with random numbers from 1 to 100. A small fitness was preferred when we evaluated individuals (permutations) of our population using both types of cost matrices.

Tables 1 and 2 show the characteristics of each of the simulated problems. For each problem, they list the number of nodes of each role and the length of the permutation that the forest represents. Note that the length of the permutation for each problem depends on the number of nodes of each role and their maximum allowed degree.

We obtained a wide variety of problems. The number of trees in the forest ranged from a single tree ($|R| = 1$, six times out of 60) to 30 trees (problem number 2 with 200 nodes and Euclidean distances). The number of intermediate nodes ranged from 25% to 70% of all

Table 2: Description of the simulated problems with random distances. The table shows the number of root, intermediate, leaf and dummy nodes and the length of the permutations that encode the problem solutions.

Random distances	$\|R\|$ **roots**	$\|I\|$ **intermediate**	$\|L\|$ **leaf**	$\|D\|$ **dummy**	n **permutation length**
Problem 1-20Nodes	2	11	7	5	23
Problem 2-20Nodes	3	10	7	9	26
Problem 3-20Nodes	3	9	8	3	20
Problem 4-20Nodes	2	9	9	5	23
Problem 5-20Nodes	1	13	6	5	24
Problem 1-40Nodes	7	12	21	3	36
Problem 2-40Nodes	7	22	11	31	64
Problem 3-40Nodes	4	12	24	1	37
Problem 4-40Nodes	4	24	12	23	59
Problem 5-40Nodes	7	10	23	3	36
Problem 1-60Nodes	1	32	27	6	65
Problem 2-60Nodes	2	42	16	30	88
Problem 3-60Nodes	1	35	24	18	77
Problem 4-60Nodes	3	41	16	34	91
Problem 5-60Nodes	11	24	25	30	79
Problem 1-80Nodes	6	43	31	31	105
Problem 2-80Nodes	5	47	28	33	108
Problem 3-80Nodes	14	32	34	24	90
Problem 4-80Nodes	2	55	23	34	112
Problem 5-80Nodes	7	44	29	22	95
Problem 1-100Nodes	11	51	38	28	117
Problem 2-100Nodes	18	42	40	47	129
Problem 3-100Nodes	7	59	34	39	132
Problem 4-100Nodes	10	63	27	58	148
Problem 5-100Nodes	12	38	50	12	100
Problem 1-200Nodes	22	98	80	62	240
Problem 2-200Nodes	16	102	82	50	234
Problem 3-200Nodes	19	99	82	51	232
Problem 4-200Nodes	1	110	89	29	228
Problem 5-200Nodes	4	122	74	60	256

network nodes. The number of leaf nodes ranged from 26% to 60% of $|V|$ depending on the problem. No dummy node had to be added in one of the problems (problem number 2 with 20 nodes and Euclidean distances), whereas problem number 2 with 40 nodes and random distances had 31 dummy nodes (77.5% of the problem size). On average, the permutation length of the problems with Euclidean distances was 15.6% greater than the number of nodes in the problem. For problems with random distances, this average rose to 22.8%. This increase is due exclusively to the fact that the problems were generated randomly.

6. Results

We used the eight algorithms described in Section 4 to solve the 60 simulated problems described in Section 5. Each problem was run 20 times with each algorithm (with a new randomly generated initial population in each run). All the results were obtained using the Magerit supercomputer. Magerit is offered by the high performance computing area at the Supercomputing and Visualization Center of Madrid (CeSViMa). Magerit is a general-purpose cluster with dual architecture (Intel and POWER). We used POWER7 nodes with 3.3 GHz (422.4 GFlops) with 32 GB of RAM and 300 GB of local hard disk.

Tables 3 and 4 show the mean and the standard deviation of the fitness value of the best solutions found for each algorithm. The best mean value for each problem is highlighted in gray. For both Euclidean and random distances, EDA and genetic algorithms (gGA and ssGA) obtained the lowest (the best) fitness in all cases. EDA is a clear winner up to problems of size 100; however, gGA achieved the best results for problems of size 200. Similarly, Tables 5 and 6 show the mean execution time and standard deviation (in minutes) for each algorithm. ssGA was the fastest on problems of size 20 and 40 and most problems of size 60. For larger problems, PSO and DE got lower execution times. Note, however, that, although they were fast, these two algorithms did not yield good solutions for the analyzed problems.

Fig. 5(a) shows the best fitness values found by EDA in the first 20 generations of a run for problem number 1 with 20 nodes and Euclidean distances. For this problem, we were interested in building a forest of three trees. Fig. 5(b)-(f) shows the trees that represent the best individuals found in generations 1, 5, 10, 15 and 20. Again, the roots are represented in green, intermediate nodes are shown in brown and leaf nodes in blue. In each forest, the edges that differ from the best forest found by the algorithm are shown in red. We observe that the number of red edges gradually disappears as the number of generations increases because the algorithm is approaching the best solution found. The forest output in generation 20 does not have any red edges because the algorithm did not improve after this generation, i.e., it provides the best fitness value found for this problem.

We applied the Friedman test to detect statistically significant differences considering the global set of algorithms (Friedman, 1937). We used the MULTIPLETEST package available at the SCI2S website[1]. We applied the Friedman test four times: once on the mean best fitness found by the eight algorithms for the 30 problems with Euclidean distances (Table 3), again on the mean execution time of the eight algorithms for the 30 problems with Euclidean distances (Table 5), and twice again on the same instances of the group of problems with random distances (Tables 4 and 6). The Friedman test rejected the null hypothesis of equality for both the fitness and execution time of Euclidean and random distances (p-value $\leq 10^{-10}$ in all cases). Once the null hypothesis of equality between all pairs of algorithms was rejected, we applied the Bergmann-Hommel procedure (Bergmann and Hommel, 1988) to find out which pairwise comparisons produced the differences.

1. http://sci2s.ugr.es/sicidm/

Table 3: Mean and standard deviation of the fitness value ($\bar{x}_{\pm s}$) for the best solutions found by each algorithm in 20 runs of each problem with Euclidean distances. The lowest (best) mean fitness for each problem is highlighted in gray.

Euclidean distances	EDA	gGA	ssGA	CMAES	DE	ElitistES	NonElitistES	PSO
Problem 1-20Nodes	$460.99_{\pm 5.19}$	$468.57_{\pm 11.97}$	$471.16_{\pm 11.13}$	$549.13_{\pm 30.15}$	$561.97_{\pm 25.81}$	$573.37_{\pm 31.18}$	$591.21_{\pm 31.87}$	$591.66_{\pm 56.79}$
Problem 2-20Nodes	$629.62_{\pm 0.00}$	$632.07_{\pm 3.94}$	$642.27_{\pm 9.91}$	$683.73_{\pm 16.72}$	$671.22_{\pm 15.06}$	$715.53_{\pm 34.99}$	$707.32_{\pm 34.23}$	$697.62_{\pm 42.59}$
Problem 3-20Nodes	$714.11_{\pm 6.60}$	$732.77_{\pm 21.35}$	$732.86_{\pm 23.42}$	$795.62_{\pm 27.75}$	$809.85_{\pm 17.96}$	$817.81_{\pm 41.46}$	$799.10_{\pm 30.96}$	$828.34_{\pm 42.77}$
Problem 4-20Nodes	$475.08_{\pm 2.18}$	$478.81_{\pm 5.78}$	$477.87_{\pm 6.00}$	$532.09_{\pm 38.85}$	$648.40_{\pm 34.10}$	$596.29_{\pm 51.93}$	$615.87_{\pm 43.32}$	$633.07_{\pm 48.70}$
Problem 5-20Nodes	$552.85_{\pm 5.26}$	$567.49_{\pm 13.90}$	$570.07_{\pm 11.13}$	$702.58_{\pm 41.25}$	$782.91_{\pm 22.33}$	$695.04_{\pm 48.69}$	$703.63_{\pm 45.28}$	$734.19_{\pm 44.31}$
Problem 1-40Nodes	$928.90_{\pm 9.51}$	$945.45_{\pm 15.09}$	$956.12_{\pm 20.09}$	$1237.15_{\pm 39.35}$	$1660.12_{\pm 35.40}$	$1245.14_{\pm 62.10}$	$1245.56_{\pm 68.77}$	$1441.12_{\pm 81.17}$
Problem 2-40Nodes	$870.71_{\pm 7.71}$	$881.01_{\pm 22.92}$	$886.97_{\pm 23.39}$	$1125.46_{\pm 44.33}$	$1548.21_{\pm 52.17}$	$1189.10_{\pm 62.80}$	$1168.29_{\pm 51.20}$	$1388.96_{\pm 75.28}$
Problem 3-40Nodes	$963.28_{\pm 14.09}$	$972.47_{\pm 20.65}$	$972.91_{\pm 19.10}$	$1352.74_{\pm 52.87}$	$1924.88_{\pm 50.30}$	$1370.71_{\pm 101.22}$	$1389.86_{\pm 81.18}$	$1675.16_{\pm 102.35}$
Problem 4-40Nodes	$827.47_{\pm 10.70}$	$839.79_{\pm 14.35}$	$844.35_{\pm 18.09}$	$1125.82_{\pm 49.15}$	$1543.02_{\pm 30.83}$	$1110.94_{\pm 50.37}$	$1157.69_{\pm 72.67}$	$1326.22_{\pm 59.47}$
Problem 5-40Nodes	$877.84_{\pm 10.68}$	$884.41_{\pm 13.69}$	$898.27_{\pm 17.14}$	$1164.23_{\pm 43.97}$	$1594.48_{\pm 30.11}$	$1186.15_{\pm 67.56}$	$1197.98_{\pm 66.26}$	$1363.30_{\pm 92.04}$
Problem 1-60Nodes	$1497.78_{\pm 22.01}$	$1510.12_{\pm 37.42}$	$1520.22_{\pm 23.70}$	$2023.22_{\pm 65.46}$	$2984.57_{\pm 53.69}$	$2057.55_{\pm 77.75}$	$2092.61_{\pm 100.78}$	$2553.75_{\pm 135.06}$
Problem 2-60Nodes	$1371.37_{\pm 36.04}$	$1420.77_{\pm 44.49}$	$1411.11_{\pm 39.48}$	$2247.22_{\pm 200.09}$	$3155.52_{\pm 48.89}$	$2057.06_{\pm 100.28}$	$2047.64_{\pm 93.69}$	$2701.65_{\pm 262.46}$
Problem 3-60Nodes	$1187.42_{\pm 8.96}$	$1188.15_{\pm 7.09}$	$1194.80_{\pm 13.54}$	$1738.09_{\pm 76.00}$	$2417.27_{\pm 46.09}$	$1624.98_{\pm 67.19}$	$1621.39_{\pm 68.12}$	$2079.24_{\pm 104.39}$
Problem 4-60Nodes	$1048.51_{\pm 10.31}$	$1046.42_{\pm 11.53}$	$1045.92_{\pm 17.11}$	$1558.62_{\pm 58.85}$	$2366.25_{\pm 43.86}$	$1597.14_{\pm 93.73}$	$1567.48_{\pm 87.83}$	$2017.42_{\pm 87.58}$
Problem 5-60Nodes	$1031.98_{\pm 10.37}$	$1042.64_{\pm 15.62}$	$1044.00_{\pm 18.29}$	$1741.26_{\pm 48.03}$	$2612.38_{\pm 40.42}$	$1644.51_{\pm 98.08}$	$1641.77_{\pm 85.77}$	$2159.81_{\pm 127.70}$
Problem 1-80Nodes	$1199.72_{\pm 12.28}$	$1211.43_{\pm 19.32}$	$1214.93_{\pm 19.03}$	$2309.58_{\pm 53.53}$	$3384.96_{\pm 56.03}$	$2101.21_{\pm 75.24}$	$2099.97_{\pm 96.66}$	$2885.96_{\pm 135.23}$
Problem 2-80Nodes	$1765.59_{\pm 22.67}$	$1748.81_{\pm 18.21}$	$1764.12_{\pm 18.60}$	$2491.55_{\pm 76.46}$	$3479.38_{\pm 56.74}$	$2371.97_{\pm 95.69}$	$2367.21_{\pm 86.80}$	$3031.89_{\pm 133.37}$
Problem 3-80Nodes	$1283.39_{\pm 17.16}$	$1283.89_{\pm 18.07}$	$1293.63_{\pm 27.91}$	$2268.18_{\pm 69.89}$	$3550.09_{\pm 67.01}$	$2124.02_{\pm 86.17}$	$2115.55_{\pm 116.89}$	$2956.60_{\pm 179.78}$
Problem 4-80Nodes	$1481.12_{\pm 25.11}$	$1485.13_{\pm 30.77}$	$1489.83_{\pm 34.92}$	$2399.71_{\pm 71.41}$	$3559.76_{\pm 55.53}$	$2288.49_{\pm 89.17}$	$2280.17_{\pm 86.28}$	$3043.38_{\pm 134.49}$
Problem 5-80Nodes	$1306.43_{\pm 15.53}$	$1298.42_{\pm 15.92}$	$1292.81_{\pm 15.48}$	$2184.25_{\pm 67.31}$	$3204.32_{\pm 49.01}$	$1985.62_{\pm 101.09}$	$2030.74_{\pm 126.11}$	$2657.66_{\pm 115.82}$
Problem 1-100Nodes	$1525.71_{\pm 24.97}$	$1533.76_{\pm 23.52}$	$1548.26_{\pm 26.53}$	$2748.73_{\pm 86.84}$	$4544.48_{\pm 51.07}$	$2685.81_{\pm 127.81}$	$2695.13_{\pm 80.30}$	$3852.29_{\pm 158.94}$
Problem 2-100Nodes	$1774.14_{\pm 24.93}$	$1753.95_{\pm 43.98}$	$1742.61_{\pm 33.26}$	$3152.66_{\pm 96.91}$	$5124.53_{\pm 62.04}$	$3109.74_{\pm 129.70}$	$3079.77_{\pm 185.62}$	$4383.51_{\pm 217.69}$
Problem 3-100Nodes	$1730.80_{\pm 40.98}$	$1719.55_{\pm 28.96}$	$1716.00_{\pm 29.09}$	$3113.16_{\pm 108.29}$	$5016.46_{\pm 43.85}$	$2896.74_{\pm 125.63}$	$2877.54_{\pm 133.61}$	$4266.76_{\pm 129.31}$
Problem 4-100Nodes	$1460.56_{\pm 16.93}$	$1445.01_{\pm 25.18}$	$1454.59_{\pm 17.52}$	$2811.28_{\pm 88.87}$	$4447.62_{\pm 78.75}$	$2587.69_{\pm 135.14}$	$2635.34_{\pm 104.78}$	$3757.36_{\pm 146.10}$
Problem 5-100Nodes	$1372.55_{\pm 19.75}$	$1385.11_{\pm 30.09}$	$1375.93_{\pm 28.76}$	$2743.15_{\pm 109.03}$	$4419.42_{\pm 65.90}$	$2581.24_{\pm 120.26}$	$2620.42_{\pm 159.02}$	$3738.77_{\pm 248.35}$
Problem 1-200Nodes	$3983.62_{\pm 131.40}$	$3065.69_{\pm 37.65}$	$3101.25_{\pm 52.28}$	$6546.09_{\pm 138.90}$	$10103.47_{\pm 77.51}$	$5730.20_{\pm 277.37}$	$5714.77_{\pm 228.00}$	$8500.70_{\pm 363.26}$
Problem 2-200Nodes	$3217.35_{\pm 151.68}$	$2397.17_{\pm 49.00}$	$2442.65_{\pm 50.18}$	$5864.41_{\pm 100.99}$	$9616.07_{\pm 88.22}$	$5385.99_{\pm 319.00}$	$5462.42_{\pm 258.06}$	$8093.95_{\pm 350.26}$
Problem 3-200Nodes	$4186.74_{\pm 230.88}$	$2922.76_{\pm 66.57}$	$2943.48_{\pm 34.44}$	$6444.62_{\pm 121.80}$	$10801.07_{\pm 63.21}$	$6117.66_{\pm 258.94}$	$6113.01_{\pm 382.20}$	$9323.62_{\pm 587.69}$
Problem 4-200Nodes	$4471.59_{\pm 237.12}$	$3101.01_{\pm 62.95}$	$3113.85_{\pm 42.11}$	$6922.55_{\pm 922.71}$	$11051.31_{\pm 54.66}$	$6082.94_{\pm 215.76}$	$6162.12_{\pm 355.29}$	$9536.97_{\pm 627.85}$
Problem 5-200Nodes	$3689.25_{\pm 150.19}$	$2766.96_{\pm 72.07}$	$2784.49_{\pm 52.03}$	$6470.60_{\pm 79.13}$	$10666.34_{\pm 75.94}$	$6018.15_{\pm 240.98}$	$6063.24_{\pm 290.45}$	$8836.16_{\pm 337.00}$

Table 4: Mean and standard deviation of the fitness value ($\bar{x}_{\pm s}$) for the best solutions found by each algorithm in 20 runs of each problem with random distances. The lowest (best) mean fitness for each problem is highlighted in gray.

Random distances	EDA	gGA	ssGA	CMAES	DE	ElitistES	NonElitistES	PSO
Problem 1-20Nodes	$241.86_{\pm 5.18}$	$258.74_{\pm 13.32}$	$263.78_{\pm 14.59}$	$340.03_{\pm 22.29}$	$403.18_{\pm 20.08}$	$357.09_{\pm 40.58}$	$346.27_{\pm 53.92}$	$377.01_{\pm 68.31}$
Problem 2-20Nodes	$214.91_{\pm 11.32}$	$222.60_{\pm 10.98}$	$224.06_{\pm 11.55}$	$319.52_{\pm 26.31}$	$366.78_{\pm 17.55}$	$354.92_{\pm 33.76}$	$359.32_{\pm 51.26}$	$362.94_{\pm 45.50}$
Problem 3-20Nodes	$210.50_{\pm 4.90}$	$216.07_{\pm 6.53}$	$216.15_{\pm 12.25}$	$267.91_{\pm 21.68}$	$307.03_{\pm 23.14}$	$309.26_{\pm 49.77}$	$303.72_{\pm 49.19}$	$316.24_{\pm 43.23}$
Problem 4-20Nodes	$172.90_{\pm 10.05}$	$178.05_{\pm 16.68}$	$172.40_{\pm 14.28}$	$294.74_{\pm 34.19}$	$354.69_{\pm 30.97}$	$267.28_{\pm 39.16}$	$263.67_{\pm 24.32}$	$294.74_{\pm 27.13}$
Problem 5-20Nodes	$161.43_{\pm 5.66}$	$178.88_{\pm 13.10}$	$184.09_{\pm 20.50}$	$311.66_{\pm 35.31}$	$363.50_{\pm 34.47}$	$292.25_{\pm 30.19}$	$312.85_{\pm 52.63}$	$305.62_{\pm 43.16}$
Problem 1-40Nodes	$271.58_{\pm 17.97}$	$287.85_{\pm 21.29}$	$302.76_{\pm 26.42}$	$455.13_{\pm 34.77}$	$907.68_{\pm 35.41}$	$531.31_{\pm 52.07}$	$523.35_{\pm 46.88}$	$709.24_{\pm 85.06}$
Problem 2-40Nodes	$182.75_{\pm 5.08}$	$196.55_{\pm 8.86}$	$199.70_{\pm 13.52}$	$546.81_{\pm 33.30}$	$961.71_{\pm 38.97}$	$532.89_{\pm 66.28}$	$527.10_{\pm 61.48}$	$734.71_{\pm 74.81}$
Problem 3-40Nodes	$298.65_{\pm 14.44}$	$331.50_{\pm 17.16}$	$341.14_{\pm 15.77}$	$601.25_{\pm 45.32}$	$1008.08_{\pm 44.92}$	$563.04_{\pm 62.98}$	$550.86_{\pm 46.58}$	$744.63_{\pm 60.87}$
Problem 4-40Nodes	$174.26_{\pm 7.11}$	$185.16_{\pm 7.59}$	$183.63_{\pm 9.40}$	$509.11_{\pm 30.52}$	$962.72_{\pm 39.70}$	$483.23_{\pm 50.98}$	$484.17_{\pm 63.38}$	$707.98_{\pm 103.91}$
Problem 5-40Nodes	$270.74_{\pm 10.00}$	$291.15_{\pm 17.94}$	$286.98_{\pm 16.31}$	$576.74_{\pm 34.94}$	$910.82_{\pm 45.97}$	$513.39_{\pm 57.20}$	$511.99_{\pm 56.08}$	$685.77_{\pm 83.97}$
Problem 1-60Nodes	$349.32_{\pm 16.54}$	$401.67_{\pm 33.18}$	$414.26_{\pm 41.55}$	$1074.74_{\pm 79.91}$	$2094.96_{\pm 63.77}$	$914.18_{\pm 96.31}$	$902.43_{\pm 109.20}$	$1565.84_{\pm 317.56}$
Problem 2-60Nodes	$229.27_{\pm 9.80}$	$247.99_{\pm 15.23}$	$248.40_{\pm 12.42}$	$899.51_{\pm 79.25}$	$2034.21_{\pm 56.25}$	$824.22_{\pm 96.65}$	$831.81_{\pm 83.31}$	$1559.76_{\pm 320.93}$
Problem 3-60Nodes	$275.12_{\pm 11.50}$	$288.37_{\pm 19.54}$	$288.66_{\pm 17.54}$	$876.76_{\pm 48.88}$	$2090.54_{\pm 58.30}$	$844.39_{\pm 97.39}$	$820.23_{\pm 88.55}$	$1509.68_{\pm 183.21}$
Problem 4-60Nodes	$228.48_{\pm 10.15}$	$241.70_{\pm 12.82}$	$241.33_{\pm 11.20}$	$896.69_{\pm 82.21}$	$1945.01_{\pm 45.21}$	$803.16_{\pm 87.70}$	$794.66_{\pm 105.41}$	$1485.17_{\pm 260.20}$
Problem 5-60Nodes	$212.33_{\pm 6.22}$	$224.96_{\pm 10.64}$	$224.24_{\pm 11.09}$	$579.58_{\pm 41.83}$	$1491.90_{\pm 39.57}$	$628.61_{\pm 55.48}$	$646.80_{\pm 82.87}$	$1085.43_{\pm 101.51}$
Problem 1-80Nodes	$312.45_{\pm 8.93}$	$322.20_{\pm 11.73}$	$324.52_{\pm 12.46}$	$1025.17_{\pm 93.17}$	$2613.59_{\pm 64.29}$	$1002.89_{\pm 87.13}$	$965.12_{\pm 104.18}$	$1932.20_{\pm 144.93}$
Problem 2-80Nodes	$285.91_{\pm 6.53}$	$301.75_{\pm 9.95}$	$303.24_{\pm 18.72}$	$1159.94_{\pm 65.75}$	$2678.84_{\pm 80.07}$	$989.62_{\pm 88.35}$	$1050.99_{\pm 141.06}$	$1864.73_{\pm 220.04}$
Problem 3-80Nodes	$291.80_{\pm 5.82}$	$305.32_{\pm 12.17}$	$305.60_{\pm 9.83}$	$804.20_{\pm 44.22}$	$2307.72_{\pm 49.13}$	$835.44_{\pm 77.34}$	$892.42_{\pm 91.03}$	$1679.06_{\pm 171.56}$
Problem 4-80Nodes	$347.69_{\pm 14.34}$	$384.93_{\pm 20.79}$	$376.70_{\pm 12.72}$	$1236.65_{\pm 50.79}$	$2934.28_{\pm 63.01}$	$1166.41_{\pm 134.54}$	$1236.17_{\pm 111.36}$	$2263.97_{\pm 419.44}$
Problem 5-80Nodes	$321.15_{\pm 12.80}$	$348.73_{\pm 15.87}$	$342.75_{\pm 15.96}$	$1068.62_{\pm 59.84}$	$2578.79_{\pm 53.89}$	$1015.96_{\pm 121.75}$	$975.67_{\pm 90.84}$	$1814.00_{\pm 308.47}$
Problem 1-100Nodes	$296.64_{\pm 7.71}$	$310.30_{\pm 12.86}$	$304.72_{\pm 10.50}$	$1142.82_{\pm 63.21}$	$3243.03_{\pm 43.57}$	$1137.47_{\pm 83.83}$	$1184.98_{\pm 130.79}$	$2177.87_{\pm 203.51}$
Problem 2-100Nodes	$246.25_{\pm 7.03}$	$265.22_{\pm 8.82}$	$262.52_{\pm 12.58}$	$992.26_{\pm 63.28}$	$2909.39_{\pm 68.81}$	$995.25_{\pm 107.03}$	$1024.73_{\pm 67.29}$	$2015.25_{\pm 181.84}$
Problem 3-100Nodes	$349.70_{\pm 12.31}$	$348.44_{\pm 12.03}$	$354.21_{\pm 10.04}$	$1268.05_{\pm 66.94}$	$3505.83_{\pm 74.56}$	$1282.11_{\pm 134.84}$	$1365.61_{\pm 114.40}$	$2517.75_{\pm 252.15}$
Problem 4-100Nodes	$267.68_{\pm 22.57}$	$265.17_{\pm 9.81}$	$273.49_{\pm 10.79}$	$1166.87_{\pm 61.52}$	$3317.99_{\pm 92.37}$	$1149.12_{\pm 119.46}$	$1223.47_{\pm 134.31}$	$2483.98_{\pm 400.61}$
Problem 5-100Nodes	$342.41_{\pm 14.30}$	$360.15_{\pm 16.38}$	$369.71_{\pm 23.50}$	$1183.28_{\pm 55.00}$	$3243.94_{\pm 68.28}$	$1099.98_{\pm 120.65}$	$1142.22_{\pm 120.75}$	$2287.92_{\pm 236.94}$
Problem 1-200Nodes	$1271.16_{\pm 87.08}$	$452.30_{\pm 14.76}$	$490.19_{\pm 17.48}$	$2220.64_{\pm 98.35}$	$7275.03_{\pm 136.34}$	$2384.50_{\pm 184.62}$	$2598.72_{\pm 198.94}$	$5553.53_{\pm 855.19}$
Problem 2-200Nodes	$1435.93_{\pm 103.93}$	$502.99_{\pm 19.35}$	$530.96_{\pm 13.75}$	$2376.65_{\pm 111.63}$	$7532.03_{\pm 93.78}$	$2521.14_{\pm 270.30}$	$2528.03_{\pm 267.82}$	$5469.27_{\pm 521.27}$
Problem 3-200Nodes	$1371.40_{\pm 91.35}$	$496.27_{\pm 23.91}$	$526.49_{\pm 13.22}$	$2405.00_{\pm 1146.40}$	$7312.68_{\pm 124.49}$	$2401.01_{\pm 217.54}$	$2516.84_{\pm 215.81}$	$5197.54_{\pm 628.02}$
Problem 4-200Nodes	$2444.46_{\pm 151.04}$	$632.74_{\pm 35.26}$	$685.40_{\pm 32.87}$	$7093.90_{\pm 2207.34}$	$8348.16_{\pm 142.54}$	$2852.08_{\pm 187.82}$	$2948.95_{\pm 263.66}$	$7570.55_{\pm 1091.33}$
Problem 5-200Nodes	$2080.21_{\pm 146.77}$	$565.23_{\pm 24.01}$	$614.61_{\pm 22.30}$	$5118.49_{\pm 2763.97}$	$8203.31_{\pm 135.53}$	$3084.57_{\pm 324.21}$	$2900.78_{\pm 185.26}$	$7248.20_{\pm 1381.91}$

Table 5: Mean and standard deviation ($\bar{x}_{\pm s}$) of the execution time (in minutes) of each algorithm in 20 runs of each problem with Euclidean distances. The shortest mean time for each problem is highlighted in gray.

Euclidean distances	EDA	gGA	ssGA	CMAES	DE	ElitistES	NonElitistES	PSO
Problem 1-20Nodes	$2.73_{\pm 1.03}$	$0.24_{\pm 0.02}$	$0.15_{\pm 0.01}$	$0.39_{\pm 0.02}$	$0.99_{\pm 0.31}$	$0.74_{\pm 0.18}$	$0.69_{\pm 0.29}$	$0.36_{\pm 0.10}$
Problem 2-20Nodes	$0.97_{\pm 0.13}$	$0.16_{\pm 0.00}$	$0.13_{\pm 0.01}$	$0.21_{\pm 0.01}$	$0.65_{\pm 0.22}$	$0.31_{\pm 0.08}$	$0.34_{\pm 0.12}$	$0.19_{\pm 0.05}$
Problem 3-20Nodes	$1.70_{\pm 0.43}$	$0.19_{\pm 0.01}$	$0.14_{\pm 0.01}$	$0.32_{\pm 0.01}$	$0.90_{\pm 0.37}$	$0.48_{\pm 0.18}$	$0.50_{\pm 0.22}$	$0.26_{\pm 0.09}$
Problem 4-20Nodes	$1.81_{\pm 0.25}$	$0.22_{\pm 0.02}$	$0.16_{\pm 0.02}$	$0.41_{\pm 0.02}$	$0.79_{\pm 0.24}$	$0.78_{\pm 0.29}$	$0.79_{\pm 0.28}$	$0.31_{\pm 0.10}$
Problem 5-20Nodes	$2.16_{\pm 0.34}$	$0.24_{\pm 0.03}$	$0.16_{\pm 0.02}$	$0.45_{\pm 0.02}$	$0.76_{\pm 0.25}$	$0.76_{\pm 0.18}$	$0.75_{\pm 0.30}$	$0.32_{\pm 0.08}$
Problem 1-40Nodes	$18.42_{\pm 5.30}$	$1.19_{\pm 0.21}$	$0.49_{\pm 0.07}$	$1.97_{\pm 0.14}$	$2.08_{\pm 0.51}$	$2.99_{\pm 1.27}$	$3.12_{\pm 1.24}$	$0.95_{\pm 0.20}$
Problem 2-40Nodes	$7.39_{\pm 1.56}$	$0.89_{\pm 0.20}$	$0.38_{\pm 0.05}$	$1.35_{\pm 0.07}$	$1.81_{\pm 0.66}$	$2.45_{\pm 0.74}$	$2.59_{\pm 1.06}$	$0.76_{\pm 0.30}$
Problem 3-40Nodes	$19.11_{\pm 5.82}$	$1.33_{\pm 0.31}$	$0.48_{\pm 0.10}$	$2.02_{\pm 0.11}$	$1.98_{\pm 0.72}$	$3.52_{\pm 0.86}$	$3.95_{\pm 1.22}$	$0.95_{\pm 0.31}$
Problem 4-40Nodes	$12.64_{\pm 3.55}$	$1.04_{\pm 0.19}$	$0.36_{\pm 0.07}$	$1.66_{\pm 0.12}$	$2.10_{\pm 0.98}$	$2.76_{\pm 0.89}$	$2.94_{\pm 0.88}$	$0.89_{\pm 0.23}$
Problem 5-40Nodes	$11.29_{\pm 3.00}$	$1.04_{\pm 0.34}$	$0.35_{\pm 0.09}$	$1.69_{\pm 0.10}$	$2.00_{\pm 0.62}$	$2.91_{\pm 0.71}$	$2.87_{\pm 0.67}$	$0.78_{\pm 0.22}$
Problem 1-60Nodes	$44.70_{\pm 11.96}$	$2.55_{\pm 0.66}$	$0.76_{\pm 0.11}$	$3.30_{\pm 0.25}$	$2.55_{\pm 1.02}$	$6.70_{\pm 2.26}$	$5.86_{\pm 1.96}$	$1.22_{\pm 0.41}$
Problem 2-60Nodes	$111.04_{\pm 35.23}$	$4.49_{\pm 1.03}$	$1.24_{\pm 0.23}$	$5.27_{\pm 0.71}$	$2.49_{\pm 0.97}$	$8.67_{\pm 2.78}$	$9.24_{\pm 2.60}$	$1.36_{\pm 0.47}$
Problem 3-60Nodes	$58.17_{\pm 16.22}$	$2.40_{\pm 0.39}$	$0.75_{\pm 0.11}$	$3.54_{\pm 0.30}$	$2.25_{\pm 0.72}$	$7.23_{\pm 2.26}$	$7.19_{\pm 2.33}$	$1.24_{\pm 0.49}$
Problem 4-60Nodes	$55.99_{\pm 16.27}$	$3.72_{\pm 0.85}$	$1.17_{\pm 0.20}$	$3.55_{\pm 0.20}$	$2.67_{\pm 1.13}$	$7.32_{\pm 2.62}$	$8.86_{\pm 3.01}$	$1.34_{\pm 0.50}$
Problem 5-60Nodes	$54.10_{\pm 17.72}$	$3.31_{\pm 0.59}$	$1.08_{\pm 0.17}$	$3.75_{\pm 0.32}$	$2.44_{\pm 0.73}$	$8.20_{\pm 2.53}$	$9.18_{\pm 2.23}$	$1.43_{\pm 0.37}$
Problem 1-80Nodes	$367.18_{\pm 85.91}$	$13.65_{\pm 2.53}$	$3.80_{\pm 0.66}$	$7.41_{\pm 0.58}$	$3.86_{\pm 1.76}$	$16.24_{\pm 5.74}$	$15.84_{\pm 4.80}$	$1.95_{\pm 0.66}$
Problem 2-80Nodes	$166.08_{\pm 30.83}$	$5.56_{\pm 0.91}$	$1.48_{\pm 0.28}$	$4.30_{\pm 0.29}$	$3.23_{\pm 0.90}$	$11.19_{\pm 3.01}$	$12.11_{\pm 3.14}$	$1.47_{\pm 0.63}$
Problem 3-80Nodes	$231.12_{\pm 54.79}$	$10.01_{\pm 2.03}$	$2.87_{\pm 0.42}$	$6.00_{\pm 0.18}$	$3.79_{\pm 1.41}$	$15.94_{\pm 5.36}$	$16.35_{\pm 4.68}$	$2.04_{\pm 0.59}$
Problem 4-80Nodes	$138.41_{\pm 40.87}$	$5.89_{\pm 1.00}$	$1.51_{\pm 0.20}$	$4.54_{\pm 0.27}$	$3.39_{\pm 0.86}$	$12.12_{\pm 3.76}$	$10.38_{\pm 3.65}$	$1.84_{\pm 0.75}$
Problem 5-80Nodes	$156.10_{\pm 50.10}$	$6.50_{\pm 1.19}$	$2.29_{\pm 0.37}$	$4.92_{\pm 0.31}$	$4.01_{\pm 1.26}$	$15.49_{\pm 4.69}$	$13.85_{\pm 5.69}$	$1.84_{\pm 0.78}$
Problem 1-100Nodes	$458.15_{\pm 61.01}$	$15.36_{\pm 3.15}$	$4.75_{\pm 1.07}$	$8.01_{\pm 0.64}$	$3.76_{\pm 1.24}$	$17.76_{\pm 5.67}$	$18.99_{\pm 4.78}$	$2.82_{\pm 1.13}$
Problem 2-100Nodes	$667.65_{\pm 175.45}$	$19.02_{\pm 4.91}$	$7.80_{\pm 1.52}$	$10.87_{\pm 0.65}$	$4.77_{\pm 1.45}$	$19.66_{\pm 6.25}$	$20.80_{\pm 6.88}$	$2.78_{\pm 1.06}$
Problem 3-100Nodes	$478.57_{\pm 122.69}$	$15.93_{\pm 3.82}$	$4.86_{\pm 0.98}$	$8.31_{\pm 0.88}$	$4.17_{\pm 1.40}$	$19.36_{\pm 5.56}$	$19.04_{\pm 5.91}$	$2.37_{\pm 0.87}$
Problem 4-100Nodes	$568.66_{\pm 81.96}$	$16.89_{\pm 3.77}$	$5.77_{\pm 1.12}$	$9.12_{\pm 0.63}$	$3.68_{\pm 1.17}$	$21.27_{\pm 6.93}$	$20.02_{\pm 5.80}$	$2.69_{\pm 0.94}$
Problem 5-100Nodes	$756.07_{\pm 201.91}$	$18.17_{\pm 4.20}$	$7.32_{\pm 1.58}$	$10.46_{\pm 0.74}$	$4.13_{\pm 1.48}$	$21.82_{\pm 7.22}$	$20.40_{\pm 8.42}$	$2.55_{\pm 0.96}$
Problem 1-200Nodes	$1851.28_{\pm 578.33}$	$58.70_{\pm 6.11}$	$52.94_{\pm 16.15}$	$50.42_{\pm 4.96}$	$5.12_{\pm 1.60}$	$41.18_{\pm 11.47}$	$44.84_{\pm 12.64}$	$8.39_{\pm 2.92}$
Problem 2-200Nodes	$2889.72_{\pm 672.79}$	$66.88_{\pm 12.41}$	$93.88_{\pm 32.80}$	$84.58_{\pm 7.54}$	$8.80_{\pm 3.14}$	$49.89_{\pm 21.56}$	$47.18_{\pm 14.24}$	$10.31_{\pm 4.06}$
Problem 3-200Nodes	$2549.65_{\pm 634.00}$	$70.65_{\pm 13.45}$	$75.92_{\pm 15.74}$	$66.17_{\pm 3.47}$	$5.82_{\pm 1.87}$	$45.85_{\pm 16.05}$	$47.73_{\pm 18.19}$	$7.76_{\pm 3.91}$
Problem 4-200Nodes	$2309.51_{\pm 549.53}$	$63.77_{\pm 10.52}$	$60.45_{\pm 17.78}$	$51.61_{\pm 10.23}$	$5.70_{\pm 1.97}$	$51.49_{\pm 15.17}$	$49.43_{\pm 18.83}$	$7.76_{\pm 3.28}$
Problem 5-200Nodes	$2487.08_{\pm 498.49}$	$68.50_{\pm 15.06}$	$76.44_{\pm 19.20}$	$75.21_{\pm 6.41}$	$6.76_{\pm 2.56}$	$47.86_{\pm 14.41}$	$47.06_{\pm 15.88}$	$10.81_{\pm 3.87}$

Table 6: Mean and standard deviation ($\bar{x}_{\pm s}$) of the execution time (in minutes) of each algorithm in 20 runs of each problem with random distances. The shortest mean time for each problem is highlighted in gray.

Random distances	EDA	gGA	ssGA	CMAES	DE	ElitistES	NonElitistES	PSO
Problem 1-20Nodes	$2.09_{\pm 0.53}$	$0.23_{\pm 0.02}$	$0.14_{\pm 0.02}$	$0.34_{\pm 0.01}$	$0.68_{\pm 0.26}$	$0.60_{\pm 0.22}$	$0.53_{\pm 0.21}$	$0.25_{\pm 0.10}$
Problem 2-20Nodes	$2.22_{\pm 0.78}$	$0.25_{\pm 0.02}$	$0.15_{\pm 0.01}$	$0.42_{\pm 0.02}$	$0.71_{\pm 0.18}$	$0.65_{\pm 0.23}$	$0.57_{\pm 0.26}$	$0.30_{\pm 0.05}$
Problem 3-20Nodes	$1.38_{\pm 0.34}$	$0.17_{\pm 0.01}$	$0.13_{\pm 0.01}$	$0.28_{\pm 0.01}$	$0.66_{\pm 0.27}$	$0.40_{\pm 0.10}$	$0.48_{\pm 0.16}$	$0.22_{\pm 0.08}$
Problem 4-20Nodes	$1.78_{\pm 0.60}$	$0.22_{\pm 0.02}$	$0.14_{\pm 0.01}$	$0.36_{\pm 0.02}$	$0.64_{\pm 0.24}$	$0.56_{\pm 0.13}$	$0.58_{\pm 0.18}$	$0.28_{\pm 0.10}$
Problem 5-20Nodes	$2.20_{\pm 0.62}$	$0.21_{\pm 0.01}$	$0.15_{\pm 0.02}$	$0.38_{\pm 0.02}$	$0.63_{\pm 0.23}$	$0.58_{\pm 0.17}$	$0.58_{\pm 0.17}$	$0.24_{\pm 0.05}$
Problem 1-40Nodes	$8.31_{\pm 3.89}$	$0.72_{\pm 0.09}$	$0.31_{\pm 0.05}$	$1.26_{\pm 0.05}$	$1.74_{\pm 0.60}$	$2.26_{\pm 0.97}$	$2.16_{\pm 0.83}$	$0.76_{\pm 0.16}$
Problem 2-40Nodes	$22.44_{\pm 7.35}$	$1.84_{\pm 0.41}$	$0.59_{\pm 0.12}$	$2.44_{\pm 0.23}$	$2.66_{\pm 0.84}$	$4.44_{\pm 1.54}$	$4.85_{\pm 1.89}$	$0.97_{\pm 0.30}$
Problem 3-40Nodes	$8.08_{\pm 1.97}$	$0.85_{\pm 0.19}$	$0.35_{\pm 0.06}$	$1.34_{\pm 0.08}$	$1.53_{\pm 0.45}$	$2.04_{\pm 0.57}$	$2.30_{\pm 0.75}$	$0.80_{\pm 0.22}$
Problem 4-40Nodes	$24.16_{\pm 9.12}$	$2.06_{\pm 0.56}$	$0.62_{\pm 0.11}$	$2.32_{\pm 0.17}$	$2.36_{\pm 0.85}$	$4.14_{\pm 1.75}$	$4.83_{\pm 1.61}$	$0.94_{\pm 0.29}$
Problem 5-40Nodes	$8.32_{\pm 2.38}$	$0.75_{\pm 0.11}$	$0.28_{\pm 0.04}$	$1.30_{\pm 0.06}$	$1.94_{\pm 0.63}$	$2.28_{\pm 0.76}$	$2.34_{\pm 0.65}$	$0.73_{\pm 0.20}$
Problem 1-60Nodes	$75.52_{\pm 14.76}$	$4.15_{\pm 1.07}$	$1.12_{\pm 0.22}$	$4.22_{\pm 0.20}$	$2.34_{\pm 0.81}$	$7.49_{\pm 2.83}$	$7.85_{\pm 2.15}$	$1.27_{\pm 0.48}$
Problem 2-60Nodes	$153.65_{\pm 20.60}$	$7.06_{\pm 1.58}$	$2.07_{\pm 0.32}$	$5.39_{\pm 0.45}$	$3.16_{\pm 0.92}$	$11.13_{\pm 3.52}$	$11.22_{\pm 3.66}$	$1.58_{\pm 0.58}$
Problem 3-60Nodes	$103.67_{\pm 21.74}$	$6.43_{\pm 1.69}$	$1.54_{\pm 0.25}$	$4.68_{\pm 0.40}$	$2.47_{\pm 0.60}$	$9.49_{\pm 2.68}$	$9.99_{\pm 3.67}$	$1.56_{\pm 0.61}$
Problem 4-60Nodes	$137.21_{\pm 21.27}$	$7.07_{\pm 1.33}$	$1.97_{\pm 0.38}$	$5.11_{\pm 0.29}$	$3.50_{\pm 1.29}$	$10.97_{\pm 3.54}$	$12.06_{\pm 5.23}$	$1.60_{\pm 0.55}$
Problem 5-60Nodes	$83.41_{\pm 20.38}$	$5.52_{\pm 1.44}$	$1.44_{\pm 0.26}$	$4.07_{\pm 0.28}$	$2.66_{\pm 0.86}$	$9.03_{\pm 2.78}$	$9.68_{\pm 3.71}$	$1.34_{\pm 0.57}$
Problem 1-80Nodes	$388.91_{\pm 65.96}$	$13.54_{\pm 3.91}$	$3.73_{\pm 0.77}$	$6.71_{\pm 0.45}$	$3.45_{\pm 1.16}$	$16.18_{\pm 5.02}$	$18.39_{\pm 5.04}$	$1.98_{\pm 0.56}$
Problem 2-80Nodes	$471.47_{\pm 104.24}$	$13.47_{\pm 2.38}$	$4.00_{\pm 0.77}$	$7.50_{\pm 0.76}$	$3.65_{\pm 1.34}$	$18.18_{\pm 8.13}$	$15.40_{\pm 6.79}$	$2.22_{\pm 0.65}$
Problem 3-80Nodes	$198.37_{\pm 30.49}$	$9.77_{\pm 2.54}$	$2.69_{\pm 0.45}$	$5.49_{\pm 0.55}$	$3.49_{\pm 1.11}$	$16.15_{\pm 5.19}$	$13.73_{\pm 3.79}$	$1.85_{\pm 0.64}$
Problem 4-80Nodes	$641.80_{\pm 139.22}$	$16.31_{\pm 4.45}$	$4.34_{\pm 0.68}$	$8.21_{\pm 0.55}$	$3.62_{\pm 0.99}$	$15.84_{\pm 4.83}$	$16.00_{\pm 7.96}$	$2.20_{\pm 0.86}$
Problem 5-80Nodes	$339.20_{\pm 56.27}$	$10.17_{\pm 2.40}$	$2.99_{\pm 0.69}$	$6.18_{\pm 0.63}$	$3.33_{\pm 1.15}$	$15.19_{\pm 7.02}$	$15.33_{\pm 4.86}$	$2.35_{\pm 0.97}$
Problem 1-100Nodes	$928.75_{\pm 126.28}$	$20.28_{\pm 5.20}$	$6.98_{\pm 1.37}$	$8.88_{\pm 0.51}$	$3.89_{\pm 1.26}$	$17.72_{\pm 5.67}$	$18.45_{\pm 5.07}$	$2.58_{\pm 1.06}$
Problem 2-100Nodes	$1006.34_{\pm 154.97}$	$17.08_{\pm 3.99}$	$7.23_{\pm 1.64}$	$9.40_{\pm 0.58}$	$3.55_{\pm 0.98}$	$21.82_{\pm 6.51}$	$18.97_{\pm 6.44}$	$2.86_{\pm 1.18}$
Problem 3-100Nodes	$1555.73_{\pm 291.76}$	$22.00_{\pm 5.08}$	$8.09_{\pm 1.01}$	$10.98_{\pm 0.76}$	$3.80_{\pm 2.02}$	$21.86_{\pm 6.56}$	$16.15_{\pm 5.63}$	$2.93_{\pm 1.01}$
Problem 4-100Nodes	$1668.46_{\pm 492.78}$	$22.61_{\pm 6.10}$	$9.00_{\pm 2.04}$	$12.46_{\pm 1.16}$	$3.16_{\pm 0.68}$	$20.74_{\pm 6.32}$	$18.31_{\pm 5.92}$	$2.87_{\pm 0.89}$
Problem 5-100Nodes	$593.13_{\pm 129.37}$	$18.20_{\pm 3.01}$	$5.09_{\pm 1.04}$	$6.95_{\pm 0.57}$	$3.47_{\pm 1.28}$	$18.77_{\pm 5.76}$	$16.80_{\pm 5.14}$	$2.59_{\pm 0.94}$
Problem 1-200Nodes	$2146.92_{\pm 391.88}$	$80.90_{\pm 18.08}$	$89.00_{\pm 21.93}$	$66.19_{\pm 5.46}$	$6.68_{\pm 2.12}$	$47.10_{\pm 14.40}$	$42.44_{\pm 12.85}$	$9.59_{\pm 5.13}$
Problem 2-200Nodes	$2135.35_{\pm 367.66}$	$80.03_{\pm 18.68}$	$82.50_{\pm 18.49}$	$65.59_{\pm 4.79}$	$7.49_{\pm 2.37}$	$47.69_{\pm 18.21}$	$48.55_{\pm 18.68}$	$11.01_{\pm 4.74}$
Problem 3-200Nodes	$2039.26_{\pm 344.44}$	$83.45_{\pm 15.47}$	$78.53_{\pm 15.96}$	$58.04_{\pm 11.15}$	$7.16_{\pm 2.75}$	$51.06_{\pm 19.41}$	$39.72_{\pm 14.19}$	$10.69_{\pm 4.27}$
Problem 4-200Nodes	$2239.56_{\pm 327.61}$	$114.55_{\pm 19.63}$	$94.83_{\pm 17.45}$	$32.23_{\pm 23.98}$	$6.17_{\pm 1.86}$	$44.96_{\pm 13.53}$	$39.43_{\pm 13.77}$	$4.50_{\pm 2.55}$
Problem 5-200Nodes	$2872.08_{\pm 625.65}$	$98.33_{\pm 16.24}$	$103.41_{\pm 20.04}$	$64.04_{\pm 35.14}$	$5.70_{\pm 1.95}$	$40.62_{\pm 14.33}$	$49.47_{\pm 12.56}$	$7.05_{\pm 5.18}$

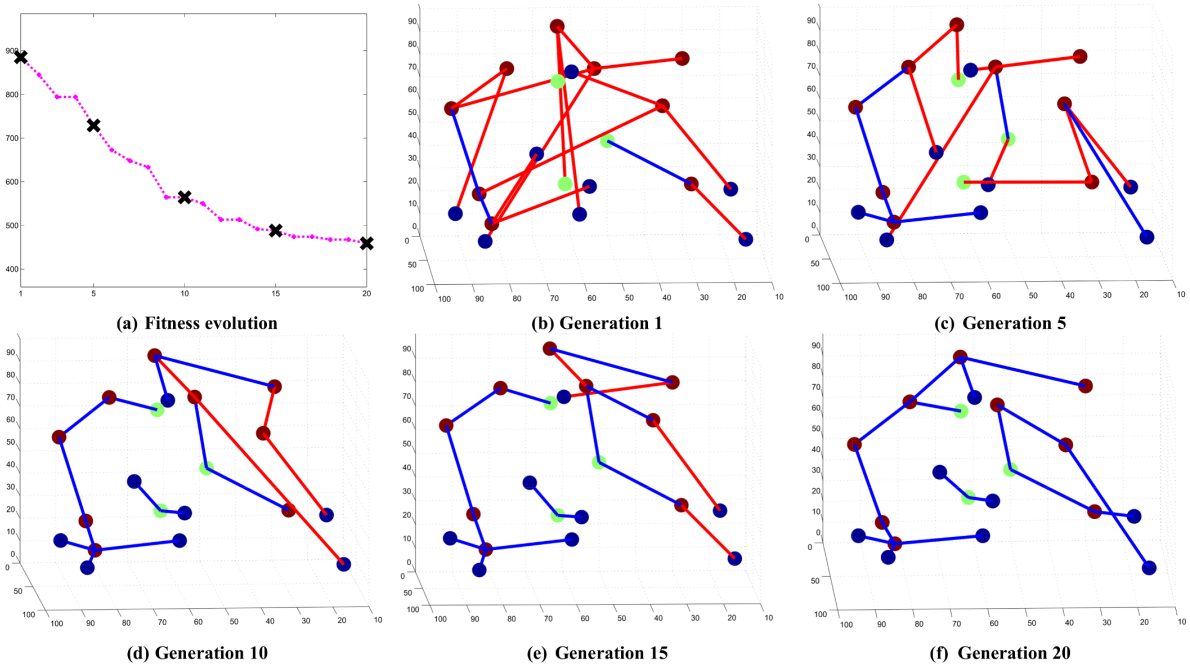


Figure 5: Evolution of the best fitness found in 20 generations by EDA for problem number 1 with 20 nodes and Euclidean distances. (a) Fitness evolution over 20 generations (the crosses indicate the fitness of individuals shown in (b)-(f)). (b)-(f) Forest encoded by the best solutions found in generations 1, 5, 10, 15 and 20. Root nodes are shown in green, intermediate nodes in brown and leaf nodes in blue. Edges that differ from the best forest found by the algorithm are shown in red. The algorithm did not improve after generation 20.

Table 7 shows the p-values of the Bergmann-Hommel procedure for all pairwise comparisons with respect to fitness and execution time for both Euclidean and random distances. p-values that do not reject equality between a pair of algorithms (p-values > 0.05) are shown in bold. Fig. 6 illustrates the results of the multiple comparison algorithms. These diagrams were introduced in Demšar (2006) and neatly illustrate statistically significant differences between algorithms. The Friedman test ranks the algorithms such that the best-performing algorithm should have rank 1, the second best rank 2, etc. In the diagrams the lowest (best) ranks are to the right so the algorithms on the right-hand side can be viewed as better. Groups of algorithms that are not significantly different are connected. Analyzing pairwise comparisons, the results showed that there were no significant differences in the best fitness for EDA, gGA and ssGA (rows 1 and 2 in columns 1 and 2 of Table 7 and diagrams at the top of Fig. 6). Looking at the execution times, however, we found significant differences between these three algorithms. EDA had a much higher execution time than both genetic algorithms for large problems (Tables 5 and 6). gGA and ssGA had similar execution times but the hypothesis of equality was rejected (row 8 in columns 3 and 4 of Table 7 and diagrams at the bottom of Fig. 6). We could therefore conclude that ssGA was preferable because it had a better execution time.

Table 7: p-values obtained in pairwise comparison algorithms using the Bergmann-Hommel procedure for fitness and execution time and Euclidean and random distances. p-values > 0.05 (equality between the two algorithms is not rejected) are shown in bold.

	Fitness		Execution time	
	Euclidean dist.	Random dist.	Euclidean dist.	Random dist.
	p-value	p-value	p-value	p-value
1 EDA vs gGA	**1.000000**	**1.000000**	1.76E-08	1.22E-06
2 EDA vs ssGA	**1.000000**	**0.641832**	1.77E-17	4.51E-16
3 EDA vs CMAES	2.73E-07	3.00E-08	1.51E-07	1.27E-08
4 EDA vs DE	1.10E-19	2.49E-23	1.22E-11	8.29E-12
5 EDA vs ElitistES	3.82E-06	1.22E-06	0.00552	0.00164
6 EDA vs NonElitistES	3.12E-06	1.20E-07	0.00552	0.00403
7 EDA vs PSO	1.11E-15	4.52E-17	8.64E-21	1.07E-21
8 gGA vs ssGA	**1.000000**	**1.000000**	0.03130	0.01096
9 gGA vs CMAES	6.25E-07	6.08E-06	**1.00000**	**0.85839**
10 gGA vs DE	6.05E-19	3.69E-19	**0.80514**	**0.28890**
11 gGA vs ElitistES	8.56E-06	9.29E-05	**0.06471**	**0.34154**
12 gGA vs NonElitistES	6.54E-06	1.89E-05	**0.06471**	**0.32543**
13 gGA vs PSO	4.66E-15	1.40E-13	0.00249	5.89E-05
14 ssGA vs CMAES	3.63E-05	1.24E-04	0.01096	**0.10624**
15 ssGA vs DE	1.11E-15	1.38E-16	**0.27507**	**0.61771**
16 ssGA vs ElitistES	2.76E-04	0.001381	6.33E-07	2.31E-05
17 ssGA vs NonElitistES	2.76E-04	2.47E-04	6.33E-07	8.30E-06
18 ssGA vs PSO	2.61E-12	1.80E-11	**1.00000**	**0.46419**
19 CMAES vs DE	0.001035	1.37E-04	**0.70008**	**0.61771**
20 CMAES vs ElitistES	**1.000000**	**1.000000**	**0.10197**	**0.07082**
21 CMAES vs NonElitistES	**1.000000**	**1.000000**	**0.10197**	**0.06471**
22 CMAES vs PSO	0.030660	0.030660	8.04E-04	0.00133
23 DE vs ElitistES	2.01E-04	9.44E-06	0.00163	0.00403
24 DE vs NonElitistES	2.01E-04	5.24E-05	0.00163	0.00201
25 DE vs PSO	**1.000000**	**0.641832**	**0.06847**	**0.06471**
26 ElitistES vs NonElitistES	**1.000000**	**1.000000**	**1.00000**	**0.85839**
27 ElitistES vs PSO	0.009759	0.005517	5.71E-09	1.12E-08
28 NonElitistES vs PSO	0.009759	0.015979	5.33E-09	2.68E-09

7. Conclusions

In this paper we have presented a novel permutation-based representation to solve a new variant of the DCMST problem, which we have called DRCMST problem. A DRCMST is a DCMST with added constraints that determine the role of the nodes in the tree (root, intermediate or leaf nodes). Establishing the roles of the nodes may be useful in some problems such as network design. Most research about computing DCMST outputs a single

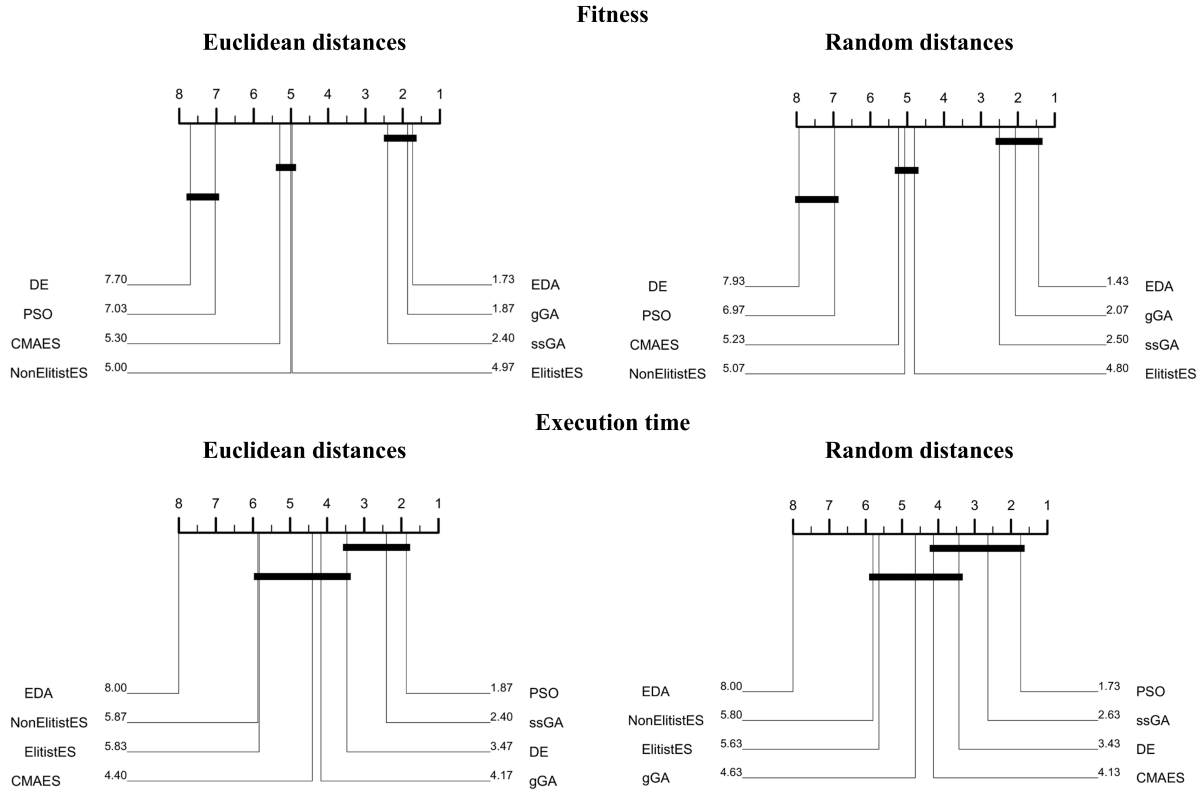


Figure 6: Comparison of the eight algorithms using the Friedman test and the Bergmann-Hommel procedure. Groups of algorithms that are not significantly different (p-value > 0.05) are connected. The lowest (best) ranks are to the right so the algorithms on the right-hand side can be viewed as better. The top row shows fitness diagrams for Euclidean distances (left) and random distances (right) and the bottom row illustrates execution time diagrams, also for Euclidean (left) and random (right) distances.

tree. We increase the flexibility of the problem by not limiting the number of root nodes to one so, generally, we compute forests of DRCMSTs. We used metaheuristic techniques to approximate the problem solution because the DRCMST problem is NP-hard. Specifically, we opted to use a range of different evolutionary computation techniques to be able to compare results. Using the proposed representation, we solved a wide range of synthetic simulated problems. The results showed that EDA and genetic algorithms (gGA and ssGA) found the best solutions, but ssGA did so in significantly less time.

The main advantage of our permutation-based representation is that it can encode more than one tree simultaneously. Moreover, the degree constraint can be different for each node. Another strength is that it is simple to add constraints related to a specific problem. For example, if two nodes cannot be connected in the problem statement, then a specific number will be forbidden at a specific position of the permutation.

Probably the weakest point of the proposed representation is that it encodes invalid individuals (cycles). Cycles of length equal to one are easy to detect and thus avoid (this

is the cause of the highest percentage of invalid individuals). However, the permutation must be decoded to detect the existence of cycles of length longer than one. Furthermore, different permutations may encode the same forest. We remove this redundancy by selecting a representative individual from the redundant individuals.

Other aspects could be taken into account such as considering a more complete fitness evaluation function. For example, if the network is designed for signal transmission from server nodes to leaf nodes, then, distances from root nodes to leaf nodes should be as short as possible, since distances are closely related to transmission time. In this case, besides minimizing the total cost (distance) of the resulting forest, it might also be beneficial to minimize the distances between roots and leaves. If there are several optimization criteria to be considered, we might also think about the convenience of optimizing either a single-objective problem (for example, weighting the different objectives) or moving towards a multi-objective problem. Another aspect to be considered is problem solving with an extremely large number of nodes. In this case, it might be handy to decompose the original problem into subproblems of smaller size and parallelize problem solving.

Genetic algorithms have been widely used to solve the DCMST problem, and some authors also use permutation representation to find the DCMST (Krishnamoorthy et al., 2001). We found that genetic algorithms also perform satisfactory for the DRCMST problem. However, EDAs have not been extensively developed for permutation-based optimization problems. We added EDAs to jMetal and found that they also performed well. We used a simple univariate EDA (Tsutsui, 2006) so it might be worthwhile to examine EDAs that provide more complex models capturing higher-order relationships between variables and analyze their performance for the DRCMST problem.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness through the Cajal Blue Brain (C080020-09; the Spanish partner of the EPFL's Blue Brain initiative) and TIN2013-41592-P projects and by the Regional Government of Madrid through the S2013/ICE-2845-CASI-CAM-CM project. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Visualization Center of Madrid (CeSViMa).

References

J.A. Aledo, J.A. Gámez, and D. Molina. Tackling the rank aggregation problem with evolutionary algorithms. *Applied Mathematics and Computation*, 222:632–644, 2013.

L. Anton-Sanchez, C. Bielza, and P. Larrañaga. Towards optimal neuronal wiring through estimation of distribution algorithms. In *Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 1647–1650, 2013.

J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *Journal on Computing*, 6(2):154–160, 1994.

B. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple Hypotheses Testing*, volume 70 of *Medizinische Informatik und Statistik*, pages 100–115. Springer Berlin Heidelberg, 1988.

C. Bielza, J.A. Fernández del Pozo, P. Larrañaga, and E. Bengoetxea. Multidimensional statistical analysis of the parameterization of a genetic algorithm for the optimal ordering of tables. *Expert Systems with Applications*, 37(1):804–815, 2010.

J. Ceberio, A. Mendiburu, and J.A. Lozano. Introducing the Mallows model on estimation of distribution algorithms. In *Neural Information Processing*, volume 7063 of *Lecture Notes in Computer Science*, pages 461–470. Springer Berlin Heidelberg, 2011.

J. Ceberio, E. Irurozki, A. Mendiburu, and J.A. Lozano. A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, 1(1):103–117, 2012.

J. Ceberio, E. Irurozki, A. Mendiburu, and J.A. Lozano. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 18(2):286–300, 2014.

H. Cobb and J. Grefenstette. Genetic algorithms for tracking changing environments. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 523–530. Morgan Kaufmann, 1993.

M.M. Czajko and J. Wojciechowski. Tree-based access network design under requirements for an aggregation network. *Elektronika - Konstrukcje, Technologie, Zastosowania*, 4: 23–27, 2009.

A.C.B. Delbem, A. de Carvalho, C.A. Policastro, A.K.O. Pinto, K. Honda, and A.C. García. Node-depth encoding for evolutionary algorithms applied to network design. In *Genetic and Evolutionary Computation*, volume 3102 of *Lecture Notes in Computer Science*, pages 678–687. Springer Berlin Heidelberg, 2004.

A.C.B. Delbem, T.W. de Lima, and G.P. Telles. Efficient forest data structure for evolutionary algorithms applied to network design. *IEEE Transactions of Evolutionary Computation*, 16(6):829–846, 2012.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

J.J. Durillo and A.J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 312–317. Morgan Kaufmann, 1996.

J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE, 1995.

J. Knowles, D. Corne, and M. Oates. A new evolutionary approach to the degree constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4: 125–134, 2000.

M. Krishnamoorthy, A.T. Ernst, and Y.M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, 7(6):587–611, 2001.

J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

P. Larrañaga and J.A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, 2002.

R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

G.R. Raidl and B.A. Julstrom. Edge sets: An effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, 2003.

I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

C.R. Reeves. A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22(1):5–13, 1995.

R. Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494, 2005.

R. Sedgewick and K. Wayne. *Algorithms, 4th Edition*. Addison-Wesley, 2011.

S. Soak, D. Corne, and B. Ahn. A new encoding for the degree constrained minimum spanning tree problem. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3213 of *Lecture Notes in Computer Science*, pages 952–958. Springer Berlin Heidelberg, 2004.

R. Storn and K. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. *Foundation of Genetic Algorithms*, pages 94–101, 1991.

S. Tsutsui. Node histogram vs. edge histogram: A comparison of probabilistic model-building genetic algorithms in permutation domains. In *IEEE Congress on Evolutionary Computation, 2006*, pages 1939–1946, 2006.